

# Föreläsning 1

- Kursinformation.
- Utvecklingsprocessen.
- Kravspecifikation.
- Gruppindelning.

# Lärare

Kursansvarig: Christer Carlsson

Gästföreläsare: Joachim von Hacht  
Michal Andreen

Handledare: Michal Andreen  
Staffan Björnesö  
Christer Carlsson  
Peter Evensen  
Daniel Wetterbro

# Kursens mål

Kursen skall ge er erfarenhet av att utveckla ett mindre programsystem i grupp, med användande av en föreskriven utvecklingsprocess.

Kursen skall också ge tillfälle att utnyttja (och därmed konsolidera) kunskaper från tidigare kurser i objektorienterad programutveckling samt människa-datorintraktion.

Ett outtalat syfte är också att ni skall ha kul under projektet.

# Undervisning

## Föreläsningar:

Måndag 16/3, kl 10-12 i sal HA2

Måndag 23/3, kl 8-10 i sal HA2

Fredag 27/3, kl 13-15 i sal HA2

Tisdag 31/3 kl 13-15 i sal HC2

## Grupphandledning:

Torsdagar i läsveckorna 1-4 samt 6-7, enligt särskilt schema. I läsvecka 5 och 8 beslutar varje grupp om handledningstid i överenskommelse med sin handledare.

*Närvaro är obligatorisk under handledningstimmarna.*

## Återkoppling:

Återkoppling ges på gruppens producerade dokument och kod.

# Deadlines

**Måndag 16/3:** Sammansättningen av projektgrupperna skall vara klart

**Onsdag 18/3:** Valet av projekt skall vara klart (senast kl 9.00)

**Fredag 20/3:** Kravspecifikation skall vara klar (senast vid midnatt).

**Fredag 27/3:** Designdokumentet skall vara klart (senast vid midnatt).

**Onsdag 13/5:** En första version av projektrapporten skall vara klar  
(senast vid midnatt).

**Tisdag 19/5:** Demoversion av programmet skall visas för handledaren.

**Tisdag 19/5:** Dokument och kod skall vara opponentgruppen tillhanda.

**Tisdag 26/5:** Projekten redovisas, all dokumentation och kod skall vara  
klar.

# Examination

## Betyg:

Endast betygen godkänd och underkänd ges.

## Krav för godkänt:

- Aktivt deltagande i gruppens arbete inklusive produktion av kod och obligatoriska dokument.
- Närvaro vid och aktivt deltagande under handledningstimmarna. Veckans arbetsinsats redovisas gruppvis skriftligen till handledaren.
- Närvaro vid och aktivt deltagande i presentationsdagen 26 maj (presentation och opposition).

# Tidsplan

| <u>Läsvecka</u> | <u>Utvecklingsfas</u> | <u>Resultat</u>                         |
|-----------------|-----------------------|-----------------------------------------|
| 1               | Kravinsamling         | Kravspecifikation                       |
| 2               | Analys och design     | Designdokument                          |
| 3-7             | Implementation        | Fungerande program,<br>manual           |
| 8               | Rapportering          | Rapport, presentation,<br>demonstration |

# Gruppindelning

Målsättningen är exakt fyra medlemmar per grupp. Om inte antalet deltagare på kursen är jämt delbart med fyra kommer några grupper att få tre medlemmar.

Gruppindelningen skall bli klar i dag.

# Resten av veckan

I dag: Gruppindelning.

Tisdag: Val av projekt.

Meddela Christer via e-post senast kl 9.00 på onsdag. Ange gruppnummer, projektval samt namn, personnummer och Chalmers CID för samtliga medlemmar).

Torsdag: Handledarmöte (se kurshemsidan för tid och plats).  
Utkast till kravspecifikation.

Fredag: Skriftlig kravspecifikation per e-post till handledaren.  
Denna och alla andra dokument skall skrivas på engelska.

# Projektförslag

Varje grupp skall välja ett av nedanstående förslag (mer information om förslagen finns på kurshemsidan).

Implementationsspråk skall i samtliga fall vara Java.

- Korsordseditor.
- Sudokueditor.
- Bibliotekssystem.
- Simulering av övergångsställe.
- System för *Lunch Dating*.
- Datorspel. Val av spel måste godkännas av handledaren.
- Eget förslag. Måste presenteras skriftligt för och godkännas av handledaren.

# Risker vid mjukvaruutveckling

- Misslyckas med att möta användarnas behov  
Gör inte det användaren vill att det ska göra.
- Misslyckas med att förutsäga kostnader  
Programsystemet blir dyrare att utveckla än förväntat.
- Misslyckas med att nå deadlines  
Är inte färdigt i tid.
- Prestandaproblem  
Är slött.
- Portabilitetsproblem  
Går inte att flytta till en annan maskin med lätthet.
- Underhållsproblem  
Har otydlig struktur och innehåller komplicerad och svårtolkad kod.
- Tillförlitlighetsproblem  
Innehåller buggar som gör att det kraschar.
- Användargränssnittsproblem  
Svårt att använda, trist layout.

# Aktiviteter vid programutveckling

Utveckling av ett större programsystem innehåller åtminstone följande aktiviteter:

- Insamlande och dokumentation av användarkrav.
- Analys av kraven.
- Design av program och GUI.
- Kodning.
- Enhetstester (tester av enskilda klasser).
- Integrationstest.

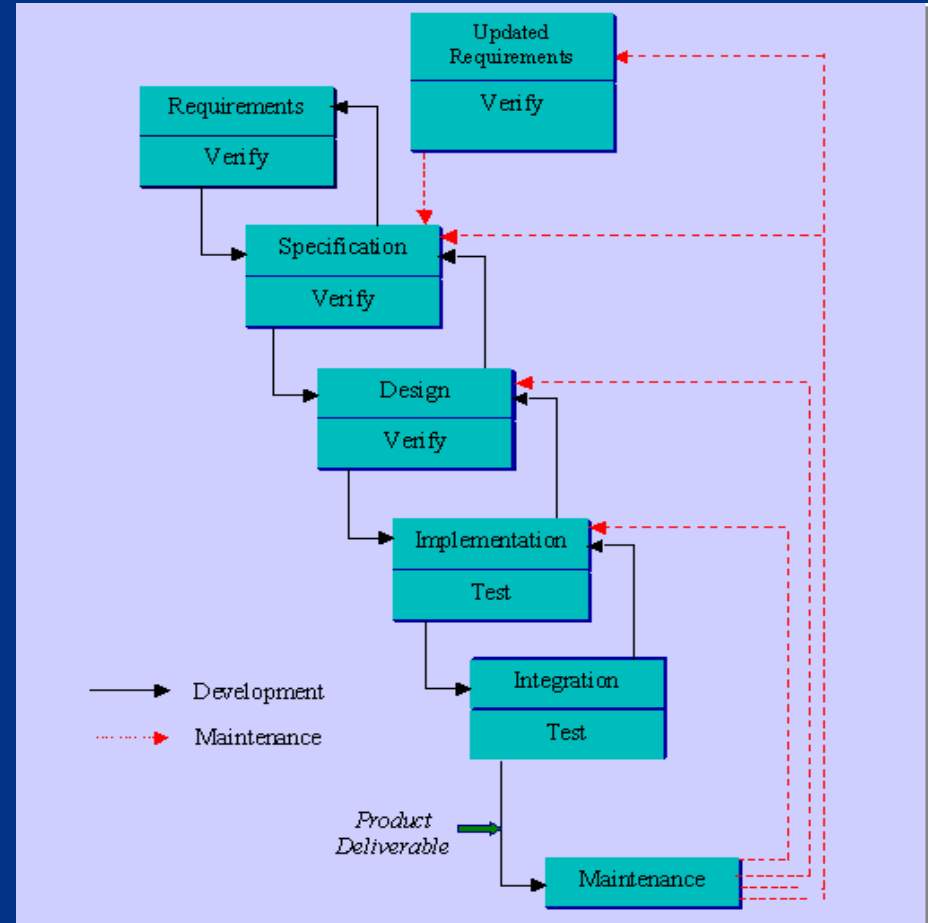
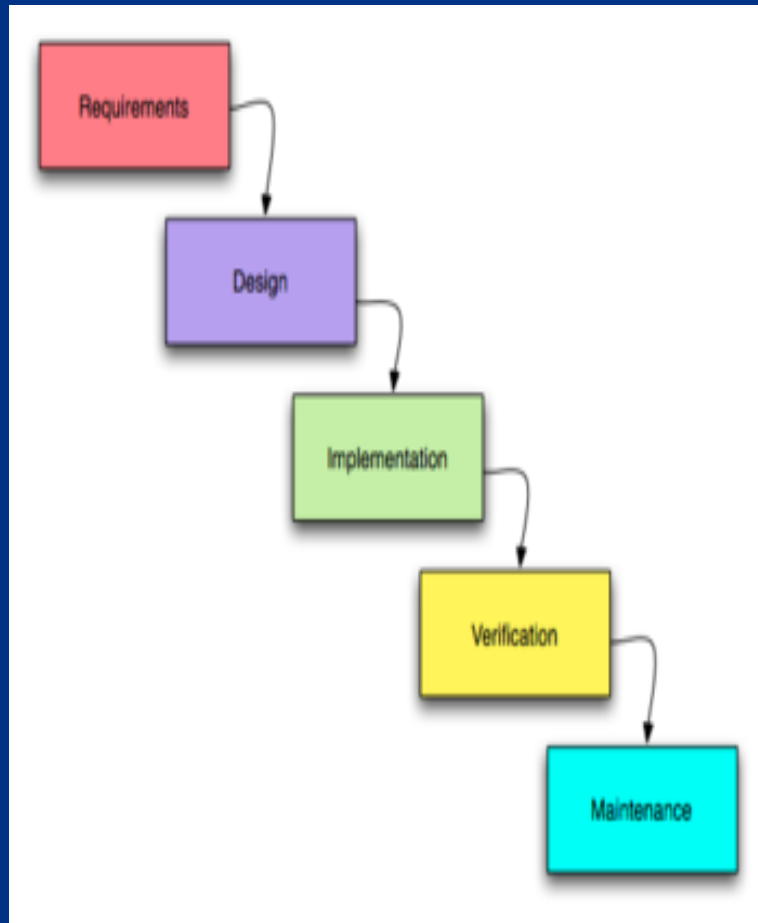
Hur skall dessa utföras och dokumenteras?

# Utvecklingsprocesser

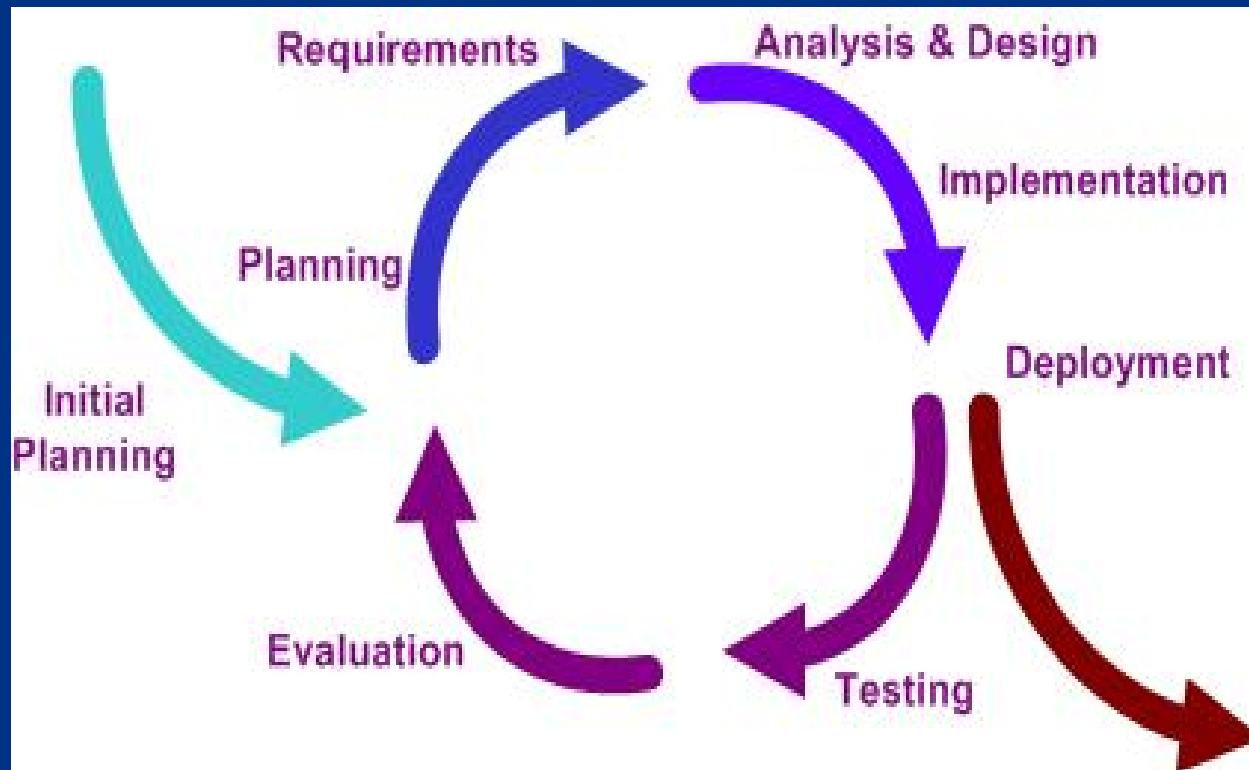
Några omtalade modeller:

- Waterfall model (1970-talet)  
Föregående fas avslutas innan nästa fas påbörjas.  
Övertagen från annan ingenjörsvetenskap.
- Iterative development (1980-talet)  
Systemet byggs i flera iterationer med utökad funktionalitet.
- Agile development (2000-talet)  
Iterativ process med korta iterationer (som längst ett par veckor) och fokus på tät samverkan i utvecklingsgrupp.

# Waterfall model (vattenfalls modellen)



# Iterative development



# Agile development

Agile development (lätttrörlig utveckling) är ett synsätt gemensamt för en grupp av utvecklingsmetoder. Det är alltså inte en utvecklingsmetodik i sig utan snarare en uppsättning värderingar, attityder och principer.

Agile värdesätter:

- **Individer och samspel** framför specifika metoder, processer och verktyg.
- **Körbar programvara** framför omfattande dokumentation.
- **Kundsamarbete** framför kontraktsförhandlingar.
- **Anpassning till förändring** framför att följa en statisk plan.

Bland lätttrörliga metoder kan nämnas: Adaptive Software Development, Crystal, DSDM, Extreme Programming (XP) och Lean Software Development.

# Vår process

- Kravbeskrivning (3 dagar).  
Enkel form av användningsfall (*use cases*).
- Analys och design (1 vecka).  
Design av GUI enligt MDI-kursen.  
Mer om design av programstrukturer nästa föreläsning.
- Implementation (4 veckor).  
I huvudsak en iteration med tillägg om tid medger.  
Enhetstest av viktiga klasser.

## Förbjudet!!

Cowboy coding: Var och en hackar där han/hon känner för det.

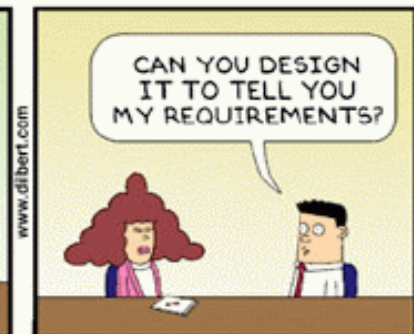
# Börja inte koda nu

Många finner det frestande att börja koda direkt, dvs:

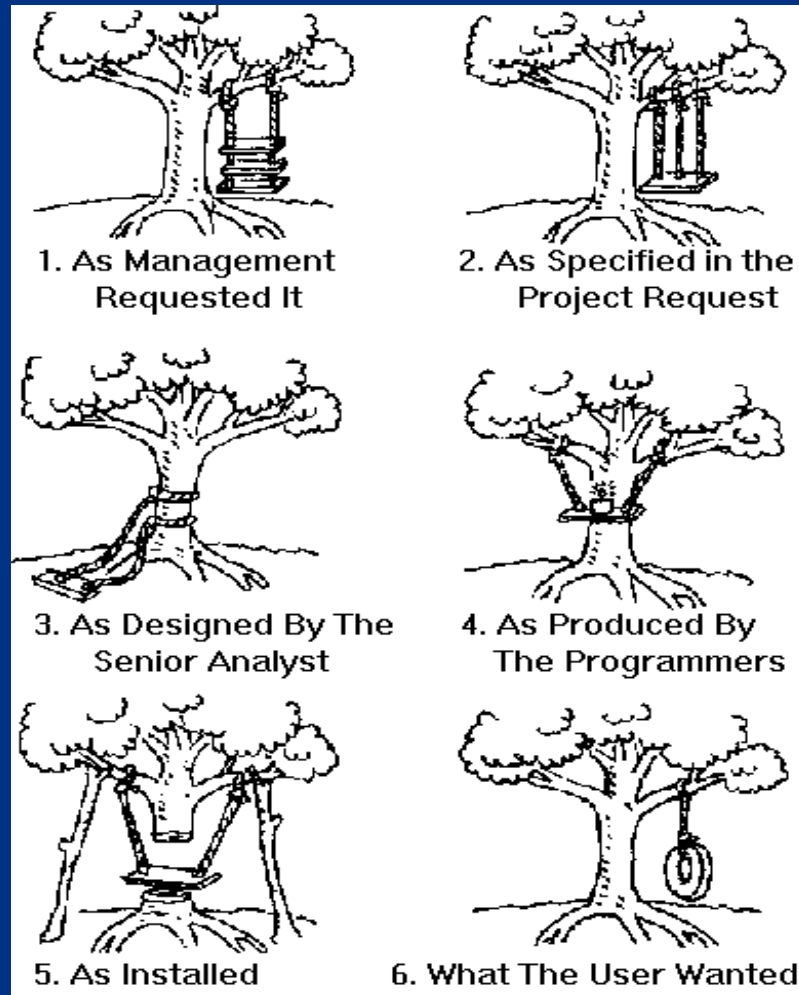
- utan att ordentligt gjort klart för sig vilka krav och önskemål kunden har,
- utan att tänka igenom en design för programmet.

Varför skulle det gå bättre vid programutveckling än inom annan ingenjörsverksamhet.

# Dilbert om kravspecifikation



# Failing to grasp the idea



# Vad är ett krav?

Ett krav är en funktion eller egenskap som systemet skall tillhandahålla för att uppfylla sitt syfte.

## Funktionella krav

Specificerar de handlingar som systemet skall kunna utföra.

## Icke-funktionella krav

Specificerar egenskaper hos systemet t.ex. användbarhet, tillförlitlighet, prestanda, modifierbarhet och återanvändbarhet.

## Nivåer av krav

**Tvingande** – absolut nödvändiga för att systemet skall uppfylla sin uppgift.

**Önskvärda** – går att klara sig utan, men detta kan leda till vissa besvär eller lägre effektivitet.

**Trevliga att ha** – bidrar endast marginellt till systemet som helhet.

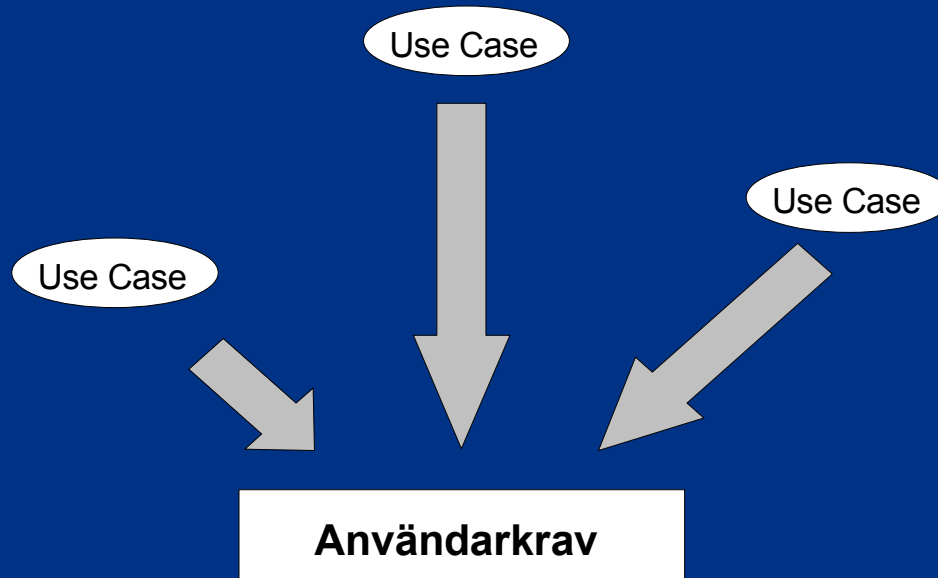
# Krav

Ett krav skall:

- specificera VAD, inte HUR
- vara korrekt
- vara komplett
- vara otvetydigt
- vara kortfattat
- vara design oberoende
- vara uppnåbart
- vara testbart

# Användningsfall (*use cases*)

Systemets funktionella krav fångas med användarfall (*use cases*). Användningsfall är en mall för en sekvens av interaktioner (scenarior) mellan en aktör (användare) och ett system



# Användningsfall (*use cases*)

Ett användarfall beskriver vilken interaktion en aktör har med systemet för att uppnå sitt mål.

En fullständig beskrivning av ett användarfall innebär ofta många steg av interaktion, med flera alternativ och felsituationer.

Vi nöjer oss med att i första skedet beskriva huvudalternativet i några få meningar. Säg i detta skede så lite som möjligt om användargränssnittet.

Vi anger för varje användarfall dess *prioritet* och *tidsåtgång*.

## Observera

Vi beskriver ett användarfall enbart med text.

Det finns grafiska standarder för att beskriva användarfall; skall dock inte användas i kursen.

# Ett exempel

Vi tänker oss ett system för att boka och sälja biljetter till en biograf med flera salonger.

Följande skall gälla för systemet:

- systemadministratörer kan lägga in föreställningar i systemet
- säljare kan boka och sälja biljetter
- ekonomisystemet kan få information om sålda biljetter.

Låt oss betrakta några användningsfall.

# Användningsfall 1

## Lägg till en film

**Aktör:** Systemadministratör.

**Mål:** Lägg till en film och dess föreställningar till systemet.

### Interaktion:

- Användaren anmodas att ge filmens namn och premiärdatum.
- Systemet visar en grafisk vy med tillgängliga visningstider för de olika salongerna den kommande månaden.
- Användaren väljer ett antal av dessa alternativ.
- Systemet bekräftar och lagrar valen.

**Prioritet:** 1.

**Svårighet:** Lätt.

# Användningsfall 2

## Sälja biljetter

**Aktör:** Säljare.

**Mål:** Sälja ett antal biljetter till en viss föreställning.

**Interaktion:**

- Användaren anger en film.
- Systemet visar en grafisk vy av salongen för nästa föreställning med lediga och upptagna platser markerade.
- Användaren kan markera ett antal lediga platser och sälja dem.

**Prioritet:** 1.

**Svårighet:** Lätt.

# Användningsfall 3

## Biljettförsäljning över internet

**Aktör:** Kund vid dator med internetanslutning.

**Mål:** Köpa biljetter till en viss föreställning.

**Interaktion:**

- Användaren anger en biograf och film.
- Systemet visar en grafisk vy av salongen för nästa föreställning med lediga och upptagna platser markerade.
- Användaren kan markera ett antal lediga platser och köpa dem och betala med kontokort.

**Prioritet:** 3.

**Svårighet:** Svår.

# Kravspecifikation som kontrakt

Den godkända kravspecifikationen fungerar som kontrakt mellan er och kunden/handledaren.

Användarfall av prioritet 1 och 2 skall implementeras.

Omprövning kan eventuellt ske i läsvecka 5, om det visar sig nödvändigt.

# Hur hitta kraven

## Ett stort problem

Vi har ingen riktig kund.

Handledaren får fungera som ersättare, men kan inte ta kundens roll på riktigt.

Förslag till arbetsmetod:

- Brainstorming (hela gruppen), för att komma på/förstå vad man skall kunna göra med programmet.
- Sortera kraven i viktiga/mindre viktiga samt lätta/svåra.
- Skriva användningsfall.

# En kravspecifikation denna vecka

## 1. Introduction

An overview of the document.

## 2. Scope

A brief description of the product to be developed.

## 3. Functional Requirements

A list of all the use cases (priority 1-3).

## 4. Other Requirements.

Time requirements? Security requirements? Data volume requirement? Etc.

## 5. User interface sketch.

Do not put too much effort into producing the sketch.